GEORGIA SOFTWORKS

RF FormMaker for Windows
NT/XP/VISTA/7/2000/2003/2008+

Fast Form Development for RF Devices

User Manual

THIS PAGE INTENTIONALLY LEFT BLANK

GEORGIA SOFTWORKS

RF FormMaker

Copyright © 1997-2011, Georgia SoftWorks, All Rights Reserved
Public Square
17 Hwy 9 South • PO Box 729
Dawsonville Georgia 30534
Telephone 706.265.1018 • Fax 706.265.1020
Visit GSW Web Site

Copyright © Georgia SoftWorks, 1997-2011 All Rights Reserved.
User's Manual, Version 1.21.0001, March 9, 2007,
Microsoft, Windows, Windows XP, Windows 2000, Windows 2003. Windows 2008 R2, Windows 7, VISTA, Windows NT, Windows 98, Windows 95 are trademarks of Microsoft Corporation. SAP, SAPConsole are trademarks of SAP AG. SecureCRT, F-Secure, PuTTY are trademarks of their respective companies.

THIS PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

LICENSOR MAKES NO WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, ORAL OR WRITTEN, REGARDING THE PROGRAM OR DOCUMENTATION AND HEREBY EXPRESSLY DISCLAIMS ALL OTHER EXPRESS OR IMPLIED WARRANTIES, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. LICENSOR DOES NOT WARRANT THE PROGRAM WILL MEET YOUR REQUIREMENTS OR THAT ITS OPERATION WILL BE UNITERRUPTED OR ERROR FREE.

IN NO EVENT WILL GEORGIA SOFTWORKS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAMS.

COPYING:

WHILE YOU ARE PERMITTED TO MAKE BACKUP COPIES OF THE SOFTWARE FOR YOU OWN USE AND PROTECTION, YOU ARE NOT PERMITTED TO MAKE COPIES FOR THE USE OF ANYONE ELSE.

LICENSE:

YOU ARE LICENSED TO RUN THIS SOFTWARE ON A SINGLE WINDOWS NT/XP/VISTA/7/2000/2003/2008+ SYSTEM. THE GEORGIA SOFTWORKS WINDOWS NT/XP/VISTA/2000/2003 SSH2 SERVER SOFTWARE MAY BE INSTALLED ON A SINGLE WINDOWS NT/XP/VISTA/7/2000/2003/2008+ SYSTEM.

This Page Left Intentionally Blank

Table of Contents

TABLE OF FIGURES III
TABLE OF TABLES III
TYPOGRAPHIC CONVENTIONS IV
FEATURES AT A GLANCE V

OVERVIEW 1

INSTALLATION 3

SERVER INSTALLATION 3
RUNTIME LICENSE 6
DEVELOPMENT LICENSE 6
FORMMAKER DEVELOPMENT 8

CONSOLESCREEN OBJECT 9

SetSize 11
SetUIElementsColor 12

CONSOLEFORM OBJECT 15

Control Placement on the Form 16
Tab Order 18
Accelerator Keys / Highlights 19
METHODS OVERVIEW 20
BUTTON CONTROL OPERATIONS 22
CreateButton 23
SetButtonColor 26
CHECKBOX OPERATIONS 28
CreateCheckBox 29
GetCheckBoxState 30
COMBOBOX OPERATIONS 31
CreateComboBox 32
AddComboBoxItem 33
SetComboBoxSelection 34
GetComboBoxSelection 35
DROPLIST OPERATIONS 36
CreateDropList 37
AddDropListItem 38
SetDropListSelection 39
GetDropListSelection 40
EDITBOX OPERATIONS 41
CreateEditBox 42
SetEditBoxText 43
SetEditBoxValidationRange 44
SetEditBoxValidationFormat 45
SetEditBoxValidationCreditCard 47
GetEditBoxText 48
LABEL OPERATIONS 49
CreateLabel 49
SetLabelColor 50
LISTBOX OPERATIONS 52
CreateListBox 53
AddListBoxItem 54

SetListBoxtSelection *55*
GetListBoxtSelection *56*
MESSAGEBOXA OPERATIONS 57
RADIOBUTTON OPERATIONS 58
CreateRadioButtonGroup *59*
AddRadioButtonGroupItem *60*
SetRadioButtonGroupSelection *61*
GetRadioButtonGroupSelection *62*
RUN METHOD 63

UTS OBJECT 64

Beep *64*
Client MAC Address *65*
Client IP Address *65*
Server Port Address *66*
Session ID *66*
TTY Name *67*
Use Color *67*
GSWClient *68*
Answerback String *68*

ERROR HANDLING 69

EXAMPLES 70

RF FORMMAKER CONTROLS 70
NAVIGATION EXAMPLE 70
DATABASE EXAMPLE 70
UTS EXAMPLE 70

REGISTRATION 71

GSW RF FORMMAKER SUBSCRIPTION 72

HOW TO UPDATE THE SOFTWARE 73
HOW TO RENEW THE GSW RF FORMMAKER SUBSCRIPTION 73

SYSTEM SIGNATURE - IMPORTANT PLEASE READ 74

TECHNICAL SUPPORT 75

PROVIDE LOG FILES TO GSW TECHNICAL SUPPORT 75

Table of Figures

Figure 1: GSW RF FormMaker Initial Setup Dialog 3
Figure 2: GSW RF FormMaker Setup Installation – Destination Location 4
Figure 3: GSW RF FormMaker Setup Installation – Program Folder 4
Figure 4: Setup Complete 5
Figure 5: UTS Program Group 5
Figure 6: New Project Template Selection 6
Figure 7: Add Reference COM selection 7
Figure 8: Typical Windows Console 9
Figure 9: ConsoleScreen Object 10
Figure 10: User Interface Elements Members 12
Figure 11: UIElements Background Colors 13
Figure 12: UIElements Foreground Colors 13
Figure 13: Color Theme 14
Figure 14 - RF FormMaker Controls, Methods and Properties 15
Figure 15: Form Control Placement 18
Figure 16: Accelerator keys 19
Figure 17 - OK or CANCEL Buttons 22
Figure 18 - Navigation Sturcture with Buttons 22
Figure 19: SetButtonColor UIElements 26
Figure 20: Background Colors 26
Figure 21: Foreground Colors 27
Figure 22: CheckBox Control 28
Figure 23: ComboBox 31
Figure 24: DropList Group 36
Figure 25: Credit Card Validation Templates 47
Figure 26: Label Background Colors 50
Figure 27: Label Foreground Colors 51
Figure 28: ListBox Group 52
Figure 29: MessageBoxA 57
Figure 30: Radio Button Group 58

Table of Tables

Table 1 - RF FormMaker Objects Usage Description 8
Table 2- Windows Console Screen and RF FormMaker Screen 10
Table 3 - Form Gird 16
Table 4 - Form Grid Coordinates Example 17
Table 5 - ConsoleForm Methods 20
Table 6 - ConsoleForm Methods Continued 21
Table 7 – Button Control Operations 22
Table 8 - CheckBox control functions 28
Table 9 - ComboBox control functions 31
Table 10 - DropList control functions 36
Table 11 - EditBox Control Functions 41
Table 12 – Label Control Operations 49
Table 13 - ListBox control functions 52
Table 14 - RadioButtonGroup control functions 58
Table 15 - UTS Properties 64
Table 16 - UTS Methods 64
Table 17: Steps to Renew the GSW Subscription Plan 73

Typographic Conventions

Italics:	are used to emphasize certain words, especially new terms or phrases when they are introduced.
Initial Caps Bold:	Words that appear in initial caps boldface represent menu options, buttons, icons or any object that you may click.
`Courier`:	This font represents anything you must type.
"<**enter**>"	This represents the enter key.

Terms/Abbreviations

UTS	GSW Universal Terminal Server
Windows	Refers to Microsoft Windows Operating Systems 98/ME/NT 4.0/XP/VISTA/2000/2003 unless otherwise noted.

Features at a Glance

Georgia SoftWorks RF FormMaker

- Abundant functions to create and process the Forms needed for your RF Application
- Build the application exactly the way you want with minimal learning curve
- Simplify your development by creating and running RF FormMaker on the Server. You do not have to learn a different development environment and implementation for every device you may encounter
- Seamlessly Integrates and works in harmony with your database software
- Easy to Install and Use
- Extend the life of your (or your customers) existing RF devices
- Produce Forms with consistent color themes
- Accelerator Keys
- Access to UTS Environment

Overview

GSW RF FormMaker is easy to install and use by the novice Visual Basic programmer, yet the capabilities and performance are appreciated by the most seasoned application and database developers.

Thank you for purchasing the Georgia SoftWorks (GSW) RF FormMaker for Windows. The RF FormMaker ActiveX object provides Visual Basic[1] programmers the ability to quickly develop forms for display on RF Devices used in Client/Server applications. With RF FormMaker the programmer can create navigation structures, labels, input/output fields and other useful controls necessary to provide an efficient high quality user interface. The colors and attributes of Form and Controls are configurable allowing a quick way of producing themes. RF FormMaker is straightforward to use and is easy to integrate with your application.

RF FormMaker greatly simplifies and speeds up application development. The time consuming, complicated, low-level work normally required for Form and Control handling for Windows Consoles is eliminated when using RF FormMaker.

RF FormMaker takes care of the creation of forms, controls, adding controls to forms, setting initial data, running the forms, and handling all user interaction with the form. When terminal events (enter key, OK, Cancel) occur then RF FormMaker provides the capability to retrieve the data of interest.

When used with GSW Mobile Clients additional features such as Stylus/Mouse, additional Function Keys, Alt Key support, file transfers, Answerback text and much more are available.

Since RF FormMaker is available for Microsoft Visual Basic, resellers and developers can rapidly create applications for their customers. Additionally companies with moderately technical staff will be able to develop applications to meet in-house needs. Testing and debugging is performed in the normal way that Visual Studio developers are familiar with.

Of course one of the main areas of development used with RF FormMaker is with databases. Often the fields on forms are populated from database queries. Additionally it is a common technique to update records based on the user input to the forms. RF FormMaker is designed to allow smooth integration with other application software, including database software.

The development and implementation of RF FormMaker takes place on the Server/Workstation. This significantly reduces the complexity of development and troubleshooting. You do not have to learn to develop on every different kind of device that you may encounter. You can take advantage of all the programming resources available on the PC, freeing you from the limitations of the device. The software is portable so if you change RF Devices the software will still run. RF FormMaker is compatible with most any RF Device that can run a Telnet or SSH2 Client.

There is no need for database synchronization between a database on the client and a database on the server since there is not a need for a database on the client. This eliminates the data transfers commonly required for synchronization that is error prone and can kill performance and network bandwidth utilization. No more (SQL-CE) synchronization nightmares!

[1] Visual Studio 2003 or Visual Studio 2005 is required for development.

RF FormMaker uses standard and familiar syntax providing help along the way. A complete running example that includes all controls is included with RF FormMaker package.

Installation

GSW RF FormMaker is fast and easy to install and register.

Server Installation

Installation of the RF FormMaker software is simple. Perform the following:

Run the setup.exe program.

1. The Welcome screen of the setup program is displayed. You are reminded and urged to exit all Windows programs before continuing. You are also reminded that you must have administrative privileges to install this program. Click **Continue.**

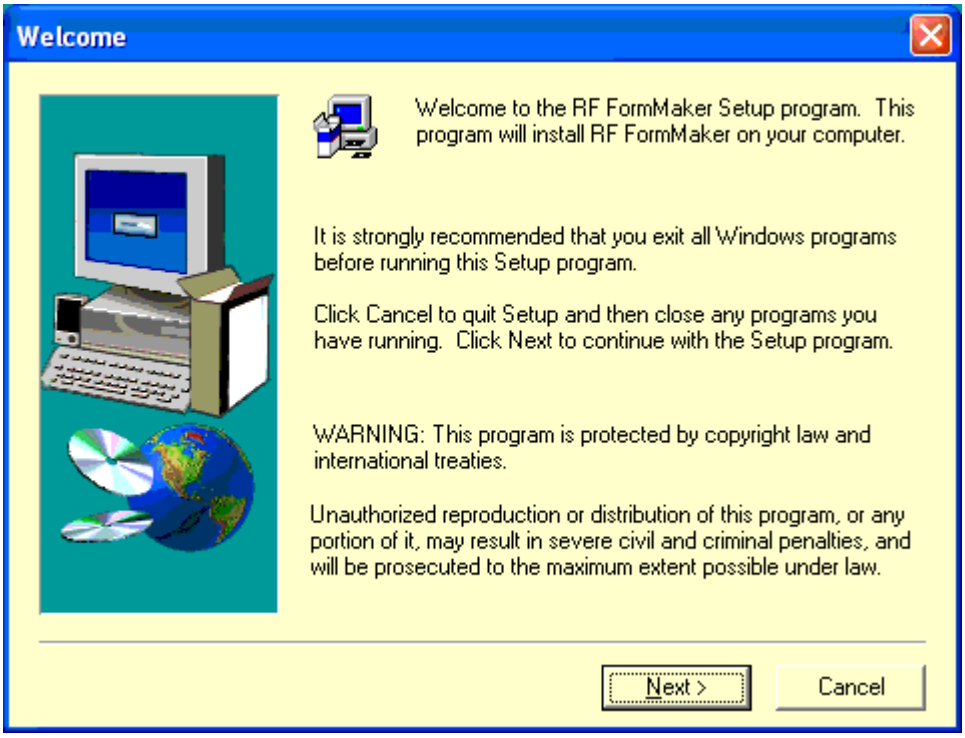
Welcome
Welcome to the RF FormMaker Setup program. This program will install RF FormMaker on your computer.
It is strongly recommended that you exit all Windows programs before running this Setup program.
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.
WARNING: This program is protected by copyright law and international treaties.
Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.
Next >
Cancel

Figure 1: GSW RF FormMaker Initial Setup Dialog

2. The Choose Destination Location dialog is displayed. You may change the destination folder if needed. Click **Next**.

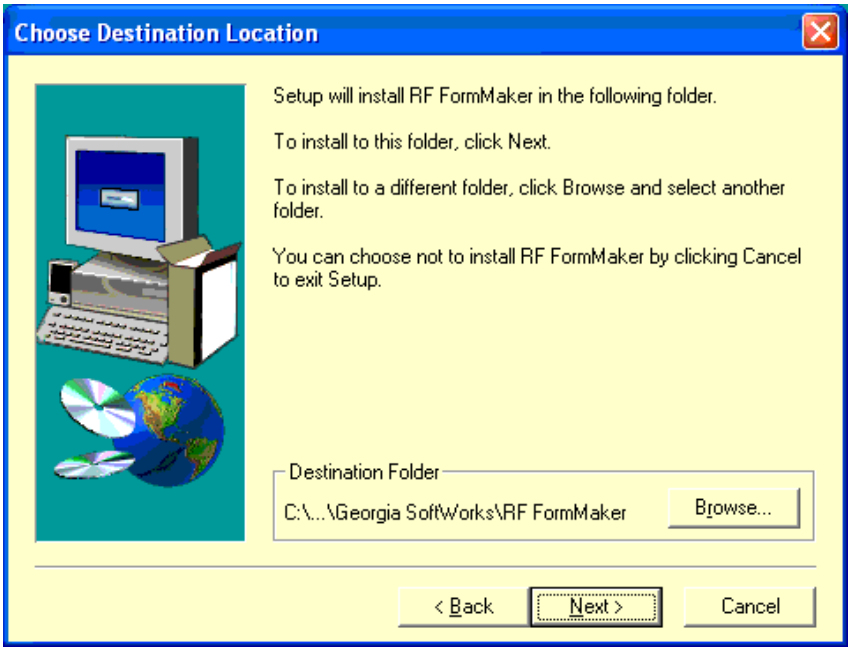
Choose Destination Location
Setup will install RF FormMaker in the following folder.
To install to this folder, click Next.
To install to a different folder, click Browse and select another folder.
You can choose not to install RF FormMaker by clicking Cancel to exit Setup.
Destination Folder
C:\...\Georgia SoftWorks\RF FormMaker
Browse...
< Back
Next >
Cancel

Figure 2: GSW RF FormMaker Setup Installation – Destination Location

3. Select Program Folder *Note: When you install on a drive that has NTFS (on the installation drive) then you must make sure that the user has full permissions to get to the installation directory and subdirectories.* Click **Next**.

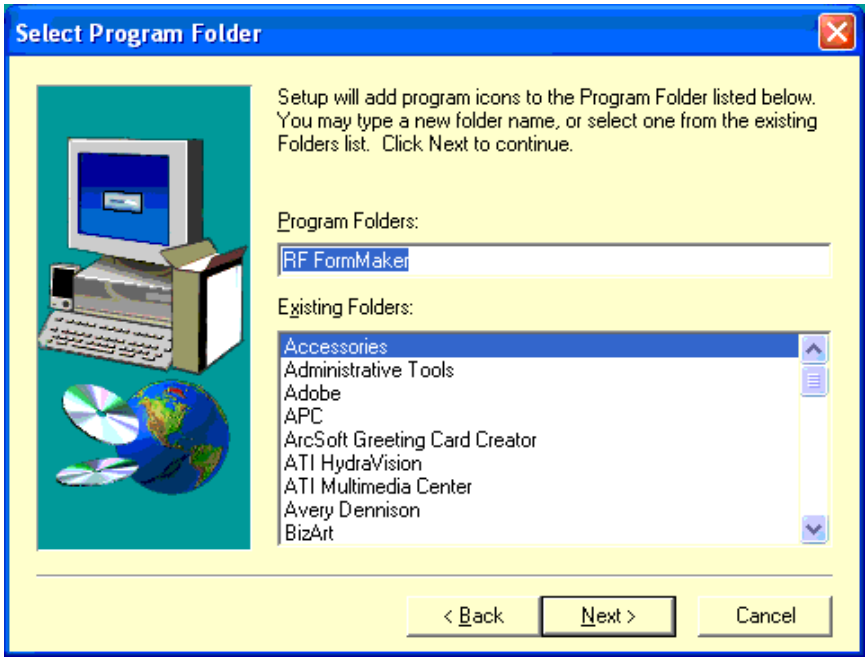

Figure 3: GSW RF FormMaker Setup Installation – Program Folder

4. A progress meter window is displayed indicating the installation progress and then installation will be complete.

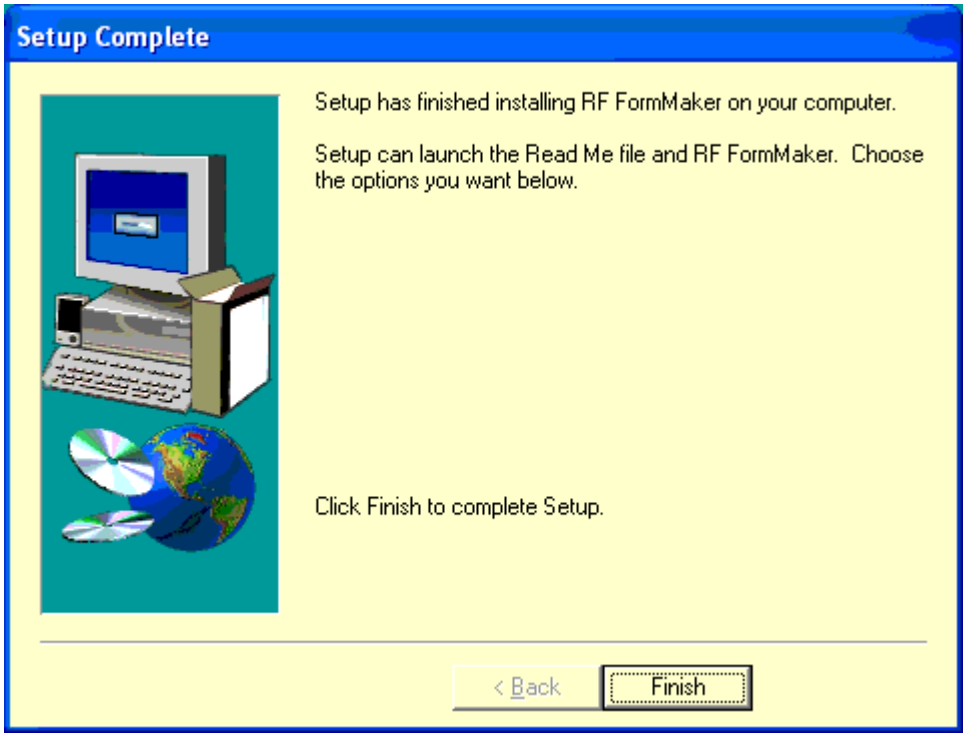
Setup Complete
Setup has finished installing RF FormMaker on your computer.
Setup can launch the Read Me file and RF FormMaker. Choose the options you want below.
Click Finish to complete Setup.
< Back
Finish

Figure 4: Setup Complete

Please view the readme.txt file as it may contain late breaking information about the telnet server that has not yet made it into the User Guide. Release notes are also contained in the Readme file.

NOTE: There are no Program Group Entries created. The RF FormMaker is transparently and automatically identified by the GSW UTS for runtime application environments as shown below in the GSW UTS Installation Status Screen.

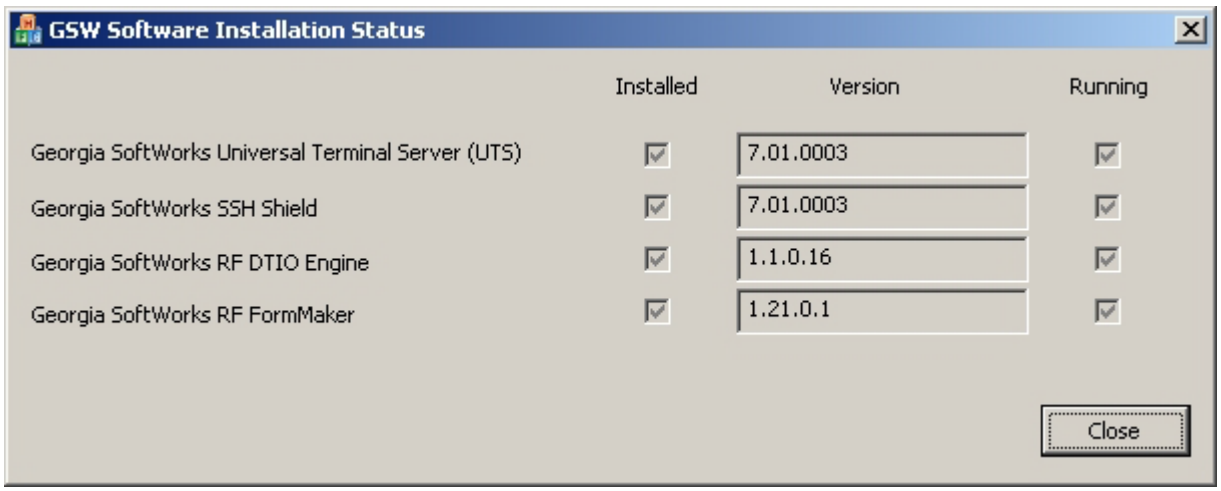

Figure 5: UTS Program Group

Two license types exist for the GSW RF FormMaker. One is for production installations and the other is for developers.

Runtime License

The GSW FormMaker must be registered in order to run in a non-development environment. The runtime license **requires** that the GSW Universal Terminal Server (UTS) is installed. Simply install the RF FormMaker and re-register the GSW UTS with the request to activate the GSW RF FormMaker.

NOTE: GSW UTS `Version 6.50.0045` or higher is required for the Runtime license of RF FormMaker.

Development License

The development license does not require a registration. It does **not** require that the GSW UTS be installed.

The development license allows use of the console screen within the development environment to test and verify correct operation. Running with the actual device's remote terminals requires a runtime license.

After installation of the software you can create a development project. In Visual Studio 2003 or higher create a new project using the Console Application Template as shown below. Choose the Names and Locations and select OK.

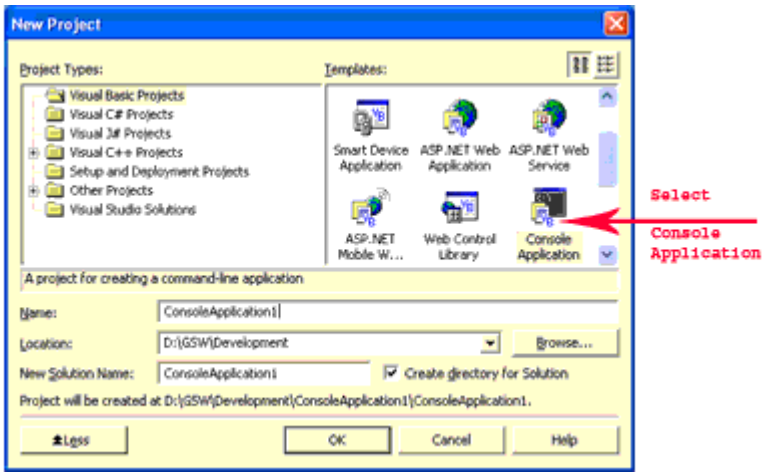

Figure 6: New Project Template Selection

Be sure to add the reference to the RF FormMaker Library in your development project.

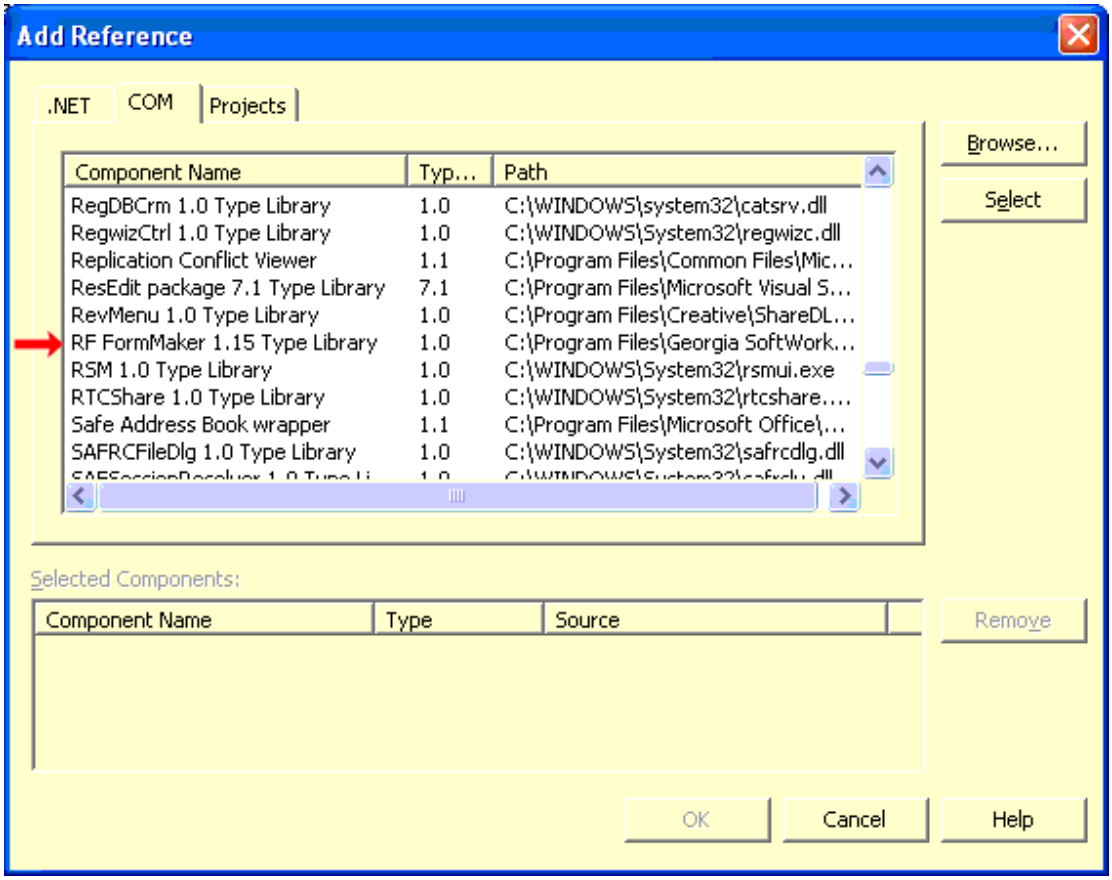

Figure 7: Add Reference COM selection

In your module be sure to add the statement

```
Imports GSWForms
```

This gives your module access to the GSW RF FormMaker controls.

In Summary the steps to get your development environment are:

1. Install RF FormMaker
2. Open Visual Studio 2003 (or higher)
3. Start a New Project using the Console Application Template
4. Add the reference to RF FormMaker Type Library
5. Add the `Imports GSWForms` statement to you module
6. Develop and test your software

FormMaker Development

Since RF Device screens come in a variety of sizes RF FormMaker allows you to define the size of the screen. The screen is similar to a background or canvas. Forms are placed on the Screen and Controls on the Forms. One screen is created per application. Multiple applications can run on a single system. The number of screens per system is determined by the license purchased and the system resources available.

All components can be dynamically created providing the programmer with unrestrained possibilities in the development process.

Development consists of creating a screen, forms and controls on the forms as well as the programming logic to process input. Program logic determines which forms are placed on the screen and when they are placed. The developer also has flexibility of controlling colors of the items on the form.

There are three objects that the developer should review to gain a better understanding of RF FormMaker. They are listed in the table below. Each object is listed in the left column of the table and the right column summarizes a description of the uses.

RF FormMaker Objects	Usage Description
RF ConsoleScreen	Sets size of RF FormMaker Console Screen
	Sets colors of User Interface Elements
RF ConsoleForm	Create controls for the form
	Add items to controls
	Set initial values for the controls on forms
	Get the values of controls on the form
	Set data validation templates
	Prompt with message box
	Run the form (Display the form on the terminal)
UTS	Beep Sound
	Read UTS Properties

Table 1 - RF FormMaker Objects Usage Description

ConsoleScreen Object

A Windows Console is an interface that provides I/O for a character mode application. RF FormMaker uses the Windows Console that is active in the context of the application.

The method that the Windows Console is created depends on the environment.

- Runtime Environment.

  The GSW UTS (Telnet/SSH2) will create the Windows Console when the session connects

- Developer Environments

  Visual Studio will create the Windows Console[2].

A typical Windows Console display is familiar and is depicted below.

Figure 8: Typical Windows Console

The size of the Windows Console is usually set using either the Windows "mode" command or from within Windows.

A GSW RF FormMaker Windows Console Screen object must be defined and created in order to have a place for your RF FormMaker forms. The code segment below defines and allocates the RF FormMaker Console Screen.

EXAMPLE 1 – DEFINE RF FORMAKER CONSOLE SCREEN

```
Dim GSWScreen As ConsoleScreen
GSWScreen = New ConsoleScreen
```

Please be aware that the Windows Console Size and the RF FormMaker Screen Size can be different. Regardless of the sizes the RF FormMaker Console Screen will reside in the top-left corner of the Windows Console when run. The RF FormMaker screen size must be the same size or smaller than the Windows Console Screen Size. This is pictured in the following table.

[2] Please review the Developer Environment section in this manual – page 7

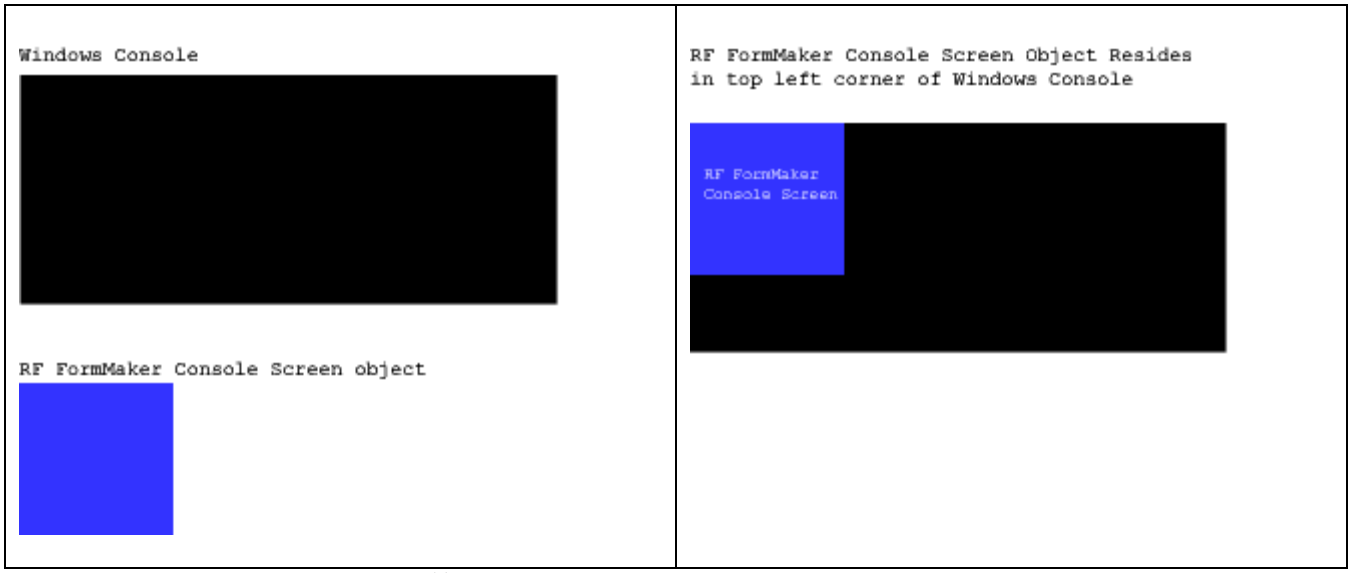
Windows Console
RF FormMaker Console Screen object
RF FormMaker Console Screen Object Resides
in top left corner of Windows Console
RF FormMaker
Console Screen

Table 2- Windows Console Screen and RF FormMaker Screen

In most cases you will want to set the RF FormMaker Screen size to match the size of the display of your RF device. There may be situations where the developer may find it beneficial to have different sizes. The developer must set the size of the RF FormMaker Console Screen and may set the color attributes of the objects placed on the screen. GSW provides good defaults for most environments while providing the developer the flexibility to customize for specific situations.

The Console Screen object has a size option as well as color attributes of the objects placed on the screen.

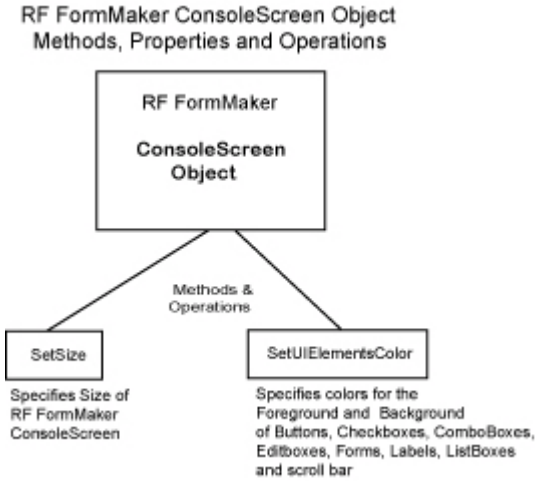

Figure 9: ConsoleScreen Object

SetSize

This method specifies the size of the RF FormMaker ConsoleScreen.

ConsoleScreen.SetSize(usLines As Short,usColumns As Short)

PARAMETERS

usLines
Required. **Short** that specifies the number of lines for the ConsoleScreen

usColumns
Required. **Short** that specifies the number of Columns for the ConsoleScreen

RETURN VALUES

EXAMPLE

```
Try
    GSWScreen = New GSWForms.ConsoleScreen
    GSWScreen.SetSize(25, 80)
Catch ex As Exception
    Console.WriteLine("Unable to create ConsoleScreen object: <" &
    ex.Message & ">")
    Return
End Try
```

SetUIElementsColor

This method specifies the background and foreground colors of all the controls, allowing a quick and easy mechanism to provide color theme presentations.

```
ConsoleScreen.SetUIElementsColor(Id As GSWForms.UIElements, Background As
GSWForms.BackgroundColors, Foreground AS GSWForms.ForegroundColors)
```

PARAMETERS

Id

Required. **GSWForms.UIElements** that allows specification for foreground and background colors for all the controls on the ConsoleScreen. This is a enumerated list where the members are:

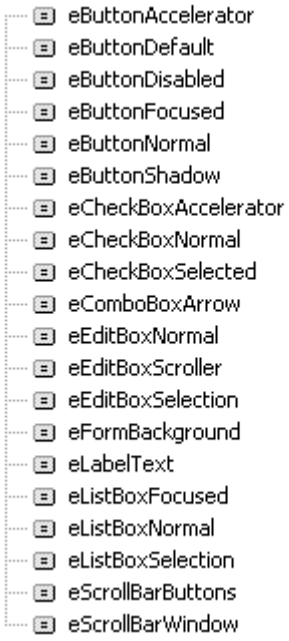

Figure 10: User Interface Elements Members

Background

Required. **GSWForms.BackgroundColors** is an enumerated list where valid values can be selected from the following list.

Figure 11: UIElements Background Colors

Foreground

Required. **GSWForms.ForegroundColors** is an enumerated list where valid values can be selected from the following list.

Figure 12: UIElements Foreground Colors

RETURN VALUES

REMARKS

EXAMPLE

```
'Blue background theme
      GSWScreen.SetUIElementColor(UIElements.eLabelText, _
      BackgroundColors.eBkgBrightCyan, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eFormBackground, _
      BackgroundColors.eBkgBrightBlue, ForegroundColors.eFgBrightRed)
GSWScreen.SetUIElementColor(UIElements.eButtonShadow, _
      BackgroundColors.eBkgBrightBlue, ForegroundColors.eFgBrightBlue)
```

```
GSWScreen.SetUIElementColor(UIElements.eButtonNormal, _
      BackgroundColors.eBkgBrightRed, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eButtonDefault, _
      BackgroundColors.eBkgBrightRed, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eButtonAccelerator, _
      BackgroundColors.eBkgBrightRed, ForegroundColors.eFgBrightWhite)
GSWScreen.SetUIElementColor(UIElements.eEditBoxNormal, _
      BackgroundColors.eBkgBrightYellow, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eListBoxNormal, _
      BackgroundColors.eBkgBrightGreen, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eScrollBarButtons, _
      BackgroundColors.eBkgBrightGreen, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eScrollBarWindow, _
      BackgroundColors.eBkgBrightGreen, ForegroundColors.eFgBlack)
GSWScreen.SetUIElementColor(UIElements.eComboBoxArrow, _
      BackgroundColors.eBkgBrightYellow, ForegroundColors.eFgBlack)
```

Figure 13: Color Theme

ConsoleForm Object

The RF ConsoleForm object is the "workhorse" form where you define the layout and controls that are displayed to the user.

The real power of RF FormMaker is the ability to create a variety of forms with controls, display the forms and interact with the user. The ConsoleForm and Controls are available to define and manage all aspects of the presentation and interaction with the user. After everything on the form is the way you want it and your program logic determines its time to display the form you "run" the form. When the form is "Run" it is placed on the RF FormMaker Console Screen, and as such is presented to the user.

Forms are created using the syntax below.

```
Dim GSWForm As ConsoleForm
GSWForm = New ConsoleForm
```

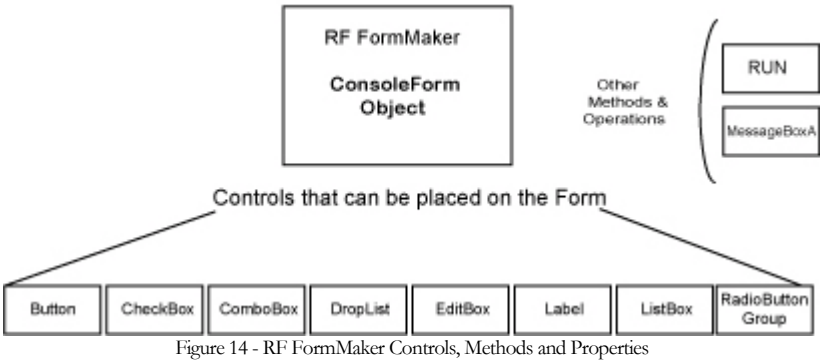

Figure 14 - RF FormMaker Controls, Methods and Properties

Each control has methods and/or properties that allow creation, configuration, interaction, placement and of the controls.

Control Placement on the Form

Conceptually, the Form can be viewed as a grid with the dimensions of the grid the same as the ConsoleScreen was defined (page 9)

The example below is if the ConsoleScreen size was:

```
GSWScreen.SetSize(16, 20)
```

Then we have a form grid of 16 rows by 20 columns. The grid row and column coordinates start with zero, where (0, 0) is the top left corner of the Form. For this document column and row numbers as well as dark borders at the edges of the form have been added for readability.

		COLUMNS																			
		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
ROWS	00																				
	01																				
	02																				
	03																				
	04																				
	05																				
	06																				
	07																				
	08																				
	09																				
	10																				
	11																				
	12																				
	13																				
	14																				
	15																				

Table 3 - Form Gird

Conceptually, each control can be viewed as contained within a rectangle. Placement of the control on the form entails defining the bounding box where the top left corner and the bottom right corner of the control should be placed on the form grid. These two points supply enough information to complete the rectangle. Coordinates are used to define each corner.

When many of the controls are created you will see the input parameters

```
TopLeftColumn,TopLeftRow,BottomRightColumn,BottomRightRow
```

These are used to specify the coordinates of the bounding box for the control.

Notice that the TopLeftRow and BottomRightRow should not be equal as then the bounding box would have no height. In general the BottomRightRow is equal to the TopLeftRow + the number of rows the control should Span. The same thought applies to the TopLeftColumn and the BottomRightColumn.

Often controls such as the Label, EditBox and Check Box will only span a single row. To some developers the BottomRightRow coordinates of the bounding box seem to cause more confusion than the others. A simple formula can be used to determine the BottomRightRow number for controls that only span a single row:

```
BottomRightRow number = TopLeftRow number + 1
```

Other controls such as the ComboBoxs, ListBoxes, DropLists and Buttons will span multiple rows.

For example, let's say we want a password label[3] prompt that we want to place on the form. The label will contain the characters PASS> and placed as show below.

		COLUMNS																			
		00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
ROWS	00																				
	01																				
	02	P	A	S	S	>															
	03																				
	04																				
	05																				
	06																				
	07																				
	08																				
	09																				
	10																				
	11																				
	12																				
	13																				
	14																				
	15																				

Table 4 - Form Grid Coordinates Example

We specify the top left corner coordinates using the TopLeftColumn and the TopLeftRow. We specify the bottom right coordinates as the BottomRightColumn and the BottomRightRow. In this example the

- TopLeftColumn is: 0.
- TopLeftRow is: 2
- BottomRightColumn is: 5
- The BottomRightRow is: 3

Remember that the row and column numbering is zero based. That is the first column is column number 0, the second column is column number 1 and so on.

[3] Please see the Label Control for further information. Page 50

The actual command to create the label will look like:

```
GSWForm.CreateLabel(0, 2, 5, 3, "PASS>")
```

And when the form is *run* the display will look something like:

Figure 15: Form Control Placement

Tab Order

Tab order will match the order that the controls are placed on the form.

Accelerator Keys / Highlights

The developer can define Accelerator keys useful for the proficient typist and for systems without mouse and stylus inputs. Even when Accelerator keys are defined, the controls can be selected by the normal means too.

Accelerator keys are characters that can be entered to select a control. The Accelerator key has a highlighted character that identifies the character. In the form below there are five buttons and each one can be selected using an Accelerator key. Shipping can be selected by pressing S, Receiving can be selected by pressing **R**, Reports can be selected by pressing e, Database can be selected by pressing **D** and the Close button can be selected by pressing **C**.

Figure 16: Accelerator keys

For all controls the highlighted accelerator character is defined using a tilde character (~). The tilde should appear before and after the desired character to use as the accelerator. Each accelerator character should be unique to a form.

The example below shows the syntax that defines the Accelerator keys as show above[4].

```
nButton = GSWForm.CreateButton(0, 4, 14, 6, "~S~hipping>", _
                                 navShipping, ButtonStyles.eNormal)
nButton = GSWForm.CreateButton(0, 6, 14, 8, "~R~eceiving>", _
                                 navReceiving, ButtonStyles.eNormal)
nButton = GSWForm.CreateButton(0, 8, 14, 10, "R~e~ports>", _
                                 navReports, ButtonStyles.eNormal)
nButton = GSWForm.CreateButton(0, 10, 14, 12, "~D~atabase>", _
                                 navDatabase, ButtonStyles.eNormal)
GSWForm.CreateButton(10, 14, 20, 16, "~C~lose", ButtonIDs.eIDCANCEL, _
                                ButtonStyles.eNormal)
```

[4] The colors in figure 16 are not represented by the example syntax.

Methods Overview

The following table shows the methods that the **ConsoleForm** object supports.

Method	Description
AddComboBoxItem	Add an Item to a ComboBox
AddDropListItem	Add an Item to a DropList
AddListBoxItem	Add an Item to a ListBox
AddRadioButtonGroupItem	Add an Item to a RadioButtonGroup
CreateButton	Create and position a Button
CreateCheckBox	Create and position a CheckBox
CreateComboBox	Create and position a ComboBox
CreateDropList	Create and position a DropList
CreateEditBox	Create and position a EditBox
CreateLabel	Create and position a Label
CreateListBox	Create and position a ListBox
CreateRadioButtonGroup	Create and position a ComboBox
GetCheckBoxState	Get/Read the value of the CheckBox (1 = True , 0 = False)
GetComboBoxSelection	Get the value selected or entered in the ComboBox
GetDropListSelection	Get/Read the value of the item selected from the DropList
GetEditBoxText	Get/Read the value entered into the EditBox
GetListBoxSelection	Get/Read the value of the item selected from the ListBox
GetRadioButtonGroupSelection	Get/Read the value of the item selected from the RadioButtonGroup

Table 5 - ConsoleForm Methods

Method	Description
MessageBoxA	Displays a message box to the user
Run	Display the form on the ConsoleScreen
SetComboBoxSelection	Set/Initialize the ComboBox value
SetDropListSelection	Set/Initialize the DropList value
SetEditBoxText	Set/Initialize the EditBox value
SetEditBoxValidationCreditCard	Set/Initialize the Credit Card validation format
SetEditBoxValidationFromat	Set/Initialize the Validation format for the EditBox
SetEditBoxValidationRange	Set/Initialize the Validation Range for the EditBox
SetListBoxSelection	Set/Initialize the ListBox value
SetRadioButtonGroupSelection	Set/Initialize an item in the RadioButtonGroup

Table 6 - ConsoleForm Methods Continued

Button Control Operations

The Button control creates and positions a Button with a text description on the form. With RF FormMaker the color of the text and the text background can be specified providing pleasant visual distinction.

The figure below shows two buttons in a common scenario where the user can click either OK or CANCEL.

Figure 17 - OK or CANCEL Buttons

Buttons are useful for many situations ranging from confirming completion of a form to creating a navigation structure as shown below.

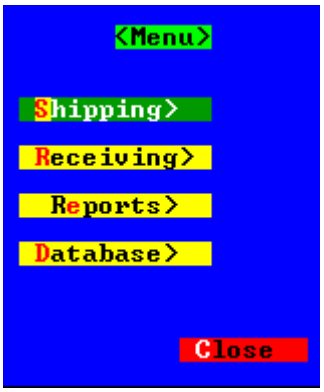

Figure 18 - Navigation Sturcture with Buttons

The figure above shows five buttons that create a navigation structure including a close button. Upon clicking/selecting a button the appropriate form will be displayed to the user

Label Control Functions		Description
`CreateButton`		Creates and defines the location and size of the button
`SetButtonColor`		Sets foreground and background colors for the button

Table 7 – Button Control Operations

When a button is clicked it returns program control to the statement following the `ConsoleForm.Run` method.

CreateButton

This method creates and defines the location of the Button.

```
ConsoleForm.CreateButton(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow as Short, strText as String, iCommand
As GSWForms.ButtonIDs, iOptions As GSWForms.ButtonStyles)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of Button.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of Button.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of Button.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of Button.

strText
Required. **String** that specifies the text displayed on the Button.

iCommand
Required. **GSWForms.ButtonIDs** that specifies the return value for the **Form** when this button is clicked.

- eIDCANCEL
- eIDNO
- eIDOK
- eIDUSERMAX - See Remarks on Page 24
- eIDUSERMIN - See Remarks on Page 24
- eIDYES

iOptions
Required. **GSWForms.ButtonStyles** that specifies if the Button is the default button when the <ENTER> key is pressed[5].

- eDefault
- eNormal

RETURN VALUES

Returns an **Integer** that is used to identify this Button when other methods are performed.

[5] Only one button per form can be designated as the eDefault Button.

REMARKS

There may be times when the standard return values:

- eIDCANCEL
- eIDNO
- eIDOK
- eIDYES

are not sufficient to address the number of buttons on the form. Two other enumerated values exist and are:

- eIDUSERMAX
- eIDUSERMIN

These values provide the developer with a mechanism to have as many buttons as needed on a form.

A set of user defined constants can be used in the iCommand parameter. The user defined constants must be in the range of eIDUSERMIN and eIDUSERMAX.

The first constant defined should be assigned the value of eIDUSERMIN. The second constant defined should be assigned the value eIDUSERMIN+1, the next eIDUSERMIN+2 and so on.

When a button is clicked the return value can be compared to the user defined constants to identify which button was clicked

The example below shows four user defined constants and how they are used.

EXAMPLE1

```
Private Const navShipping As Integer = GSWForms.ButtonIDs.eIDUSERMIN
Private Const navReceiving As Integer = GSWForms.ButtonIDs.eIDUSERMIN + 1
Private Const navReports As Integer = GSWForms.ButtonIDs.eIDUSERMIN + 2
Private Const navDatabase As Integer = GSWForms.ButtonIDs.eIDUSERMIN + 3


  nButton = GSWForm.CreateButton(0, 4, 14, 6, "~S~hipping>", _
                                 navShipping, ButtonStyles.eNormal)
  nButton = GSWForm.CreateButton(0, 6, 14, 8, "~R~eceiving>", _
                                 navReceiving, ButtonStyles.eNormal)
  nButton = GSWForm.CreateButton(0, 8, 14, 10, "R~e~ports>", _
                                 navReports, ButtonStyles.eNormal)
  nButton = GSWForm.CreateButton(0, 10, 14, 12, "~D~atabase>", _
                                 navDatabase, ButtonStyles.eNormal)
  nFormResult = GSWForm.Run()
  If nFormResult = GSWForms.ButtonIDs.eIDCANCEL Then
      Return
  End If
  Select Case nFormResult
      Case navShipping
          GSWForm.MessageBoxA("'Shipping' selected")
      Case navReceiving
          GSWForm.MessageBoxA("'Receiving' selected")
```

```
        Case navReports
            GSWForm.MessageBoxA("'Reports' selected")
        Case navDatabase
            GSWForm.MessageBoxA("'Database' selected")
        Case Else
            GSWForm.MessageBoxA("ERROR: Unexpected selection!")
    End Select
```

EXAMPLE2

This is another CreateButton example.

```
Dim nButton1 As Integer
nButton1 = GSWForm.CreateButton(2, 16, 13, 18, "~O~K", _
GSWForms.ButtonIDs.eIDOK, GSWForms.ButtonStyles.eDefault)
```

SetButtonColor

This method sets the foreground and background colors for the Button.

```
ConsoleForm.SetButtonColor(iId As Integer, iUIElementsID As GSWForms.UIElements,
Background As GSWForms.BackgroundColors, Foreground As GSWForms.ForegroundColors)
```

PARAMETERS

iId

Required. **Integer** that specifies the Button to perform the SetButtonColor method. The Identifier is returned from the CreateButton method

iUIElementIDs

Required. The button subset of **GSWForms.UIElements** that allows specification for foreground and background colors for the button. This subset is an enumerated list where the members are:

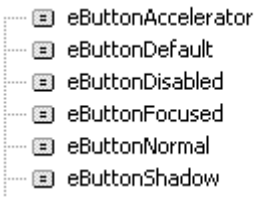

Figure 19: SetButtonColor UIElements

Background

Required. **GSWForms.BackgroundColors** is an enumerated list were valid values can be selected from the following list.

Figure 20: Background Colors

Foreground

Required. **GSWForms.ForegroundColors** is an enumerated list were valid values can be selected from the following list.

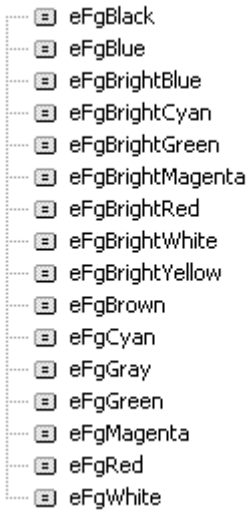

Figure 21: Foreground Colors

RETURN VALUES

EXAMPLE

```
nButton = GSWForm.CreateButton(0, 4, 14, 6, "~S~hipping>", navShipping, _
            ButtonStyles.eNormal)
GSWForm.SetButtonColor(nButton, UIElements.eButtonNormal, _
            BackgroundColors.eBkgBrightYellow, ForegroundColors.eFgBlack)
```

CheckBox Operations

A set of operations handle the creation, placement, initialization and interaction of the CheckBox. The CheckBox control is a two-state graphical control that allows the user to check or uncheck (toggle) the value. Often checkboxes are grouped together, providing a similar appearance as a radio button. The CheckBox functionality differs from the RadioButton in that checkboxes are not mutually exclusive. Grouping of checkboxes is simply due to the placement on the screen because there is no relationship enforced between checkboxes. Any and all check boxes may be checked or unchecked. The RadioButton only allows a single choice of the group.

The CheckBox example below uses three CheckBox controls grouped together vertically. However you may have a single or multiple Checkbox controls. You may place each CheckBox control independent of any other as you can see below

.

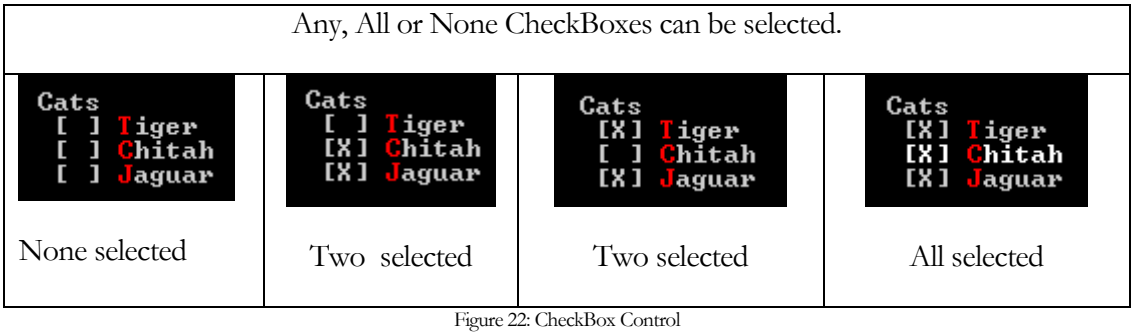

Figure 22: CheckBox Control

CheckBox Control Functions		Description
CreateCheckBox		Creates and defines the location and size of the CheckBox, text and initial value.
GetCheckBoxState		Reads the value of the Checkbox.

Table 8 - CheckBox control functions

CreateCheckBox

This operation creates and places a CheckBox on the GSWForm

ConsoleForm.**CreateCheckBox**(us*TopLeftCol As Short,usTopLeftRow As Short, usBottomRightCol As Short, usBottomRightRow As Short, strText as String, bChecked as Integer*)As Integer

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of CheckBox.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of CheckBox.

usBottomRightCol
Required. **Sthort** that specifies the Cell Coordinates of bottom right column of CheckBox.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of CheckBox.

strText
Required. **String** that specifies the text for the checkbox label

bChecked
Required. **Integer** Initial value for checkbox (True = checked, False = cleared)

RETURN VALUES

Returns an **Integer** that is used to identify this checkbox when other methods are performed.

REMARKS

EXAMPLE

The following code segment shows how to create a checkbox using the **CreateCheckBox** method.

```
Dim nCheckBox1 As Integer
Dim nCheckBox2 As Integer
Dim nCheckBox3 As Integer

nCheckBox1 = GSWForm.CreateCheckBox(2, 3, 14, 4, "~T~iger", False)
nCheckBox2 = GSWForm.CreateCheckBox(2, 4, 14, 5, "~C~hitah", True)
nCheckBox3 = GSWForm.CreateCheckBox(2, 5, 14, 6, "~J~aguar", True)
GSWForm.CreateLabel(2, 2, 14, 3, "Cats")
```

GetCheckBoxState

This method gets/reads the value of the CheckBox.

```
ConsoleForm.GetCheckBoxState(iId as Short)As Integer
```

PARAMETERS

iId

Required. **Short** identifier that specifies the CheckBox to perform the GetCheckBoxState method. The Identifier is returned from the CreateCheckBox method.

RETURN VALUES

Integer state value of the CheckBox *NonZero = Selected, Zero = Not Selected*

REMARKS

EXAMPLE

The following code segment shows how to get the value of the CheckBox using the **GetCheckBoxState** method.

```
Dim nCheckBox1 As Integer
Dim nCheckBox2 As Integer
Dim nCheckBox3 As Integer

nCheckBox1 = GSWForm.CreateCheckBox(2, 3, 14, 4, "~T~iger", False)
nCheckBox2 = GSWForm.CreateCheckBox(2, 4, 14, 5, "~C~hitah", True)
nCheckBox3 = GSWForm.CreateCheckBox(2, 5, 14, 6, "~J~aguar", True)
GSWForm.CreateLabel(2, 2, 14, 3, "Cats")
Dim nFormResult As Integer
nFormResult = GSWForm.Run()
  If nFormResult = GSWForms.ButtonIDs.eIDCANCEL Then Return

Dim nCheckBox1State As Boolean
Dim nCheckBox2State As Boolean
Dim nCheckBox3State As Boolean

nCheckBox1State = GSWForm.GetCheckBoxState(nCheckBox1)
nCheckBox2State = GSWForm.GetCheckBoxState(nCheckBox2)
nCheckBox3State = GSWForm.GetCheckBoxState(nCheckBox3)
```

ComboBox Operations

The ComboBox control is a type of graphical interface that allows the user to select exactly one item from a predefined list or enter an unanticipated value.

The ComboBox is useful when you do not want to permanently dedicate screen space to display the list contents. When the down arrow is clicked the entire list is displayed. If necessary the list may scroll if the number of items is greater than what will fit into the defined area.

The difference between a ComboBox and a DropList is that the ComboBox allows a new value to be entered in addition to the items listed.

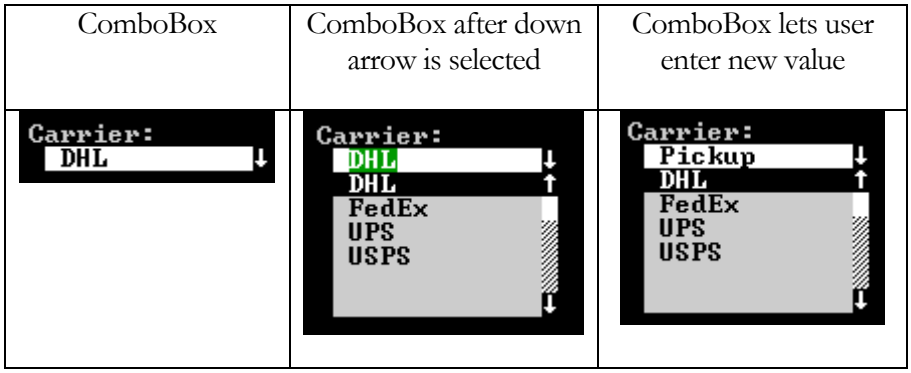

Figure 23: ComboBox

The developer may initialize an item as selected. The ComboBox control will return the string value of item selected or entered.

DropList Control Functions		Description
CreateComboBox		Creates and defines the location and size of the ComboBox
AddComboBoxItem		Adds items to the ComboBox
SetComboBoxSelection		Sets/Initializes an item in the ComboBox
GetComboBoxSelection		Reads the value of the ComboBox

Table 9 - ComboBox control functions

CreateComboBox

This method creates and places a ComboBox on the GSWForm.

```
ConsoleForm.CreateComboBox(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow As Short)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of ComboBox.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of ComboBox.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of ComboBox.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of ComboBox.

RETURN VALUES

Returns an **Integer** that is used to identify this combox when other methods are performed.

REMARKS

EXAMPLE

The following code segment shows how to create a ComboBox using the **CreateComboBox** method.

```
Dim nComboBox As Integer
nComboBox = GSWForm.CreateComboBox(38, 3, 52, 4)

GSWForm.AddComboBoxItem(nComboBox, "FedEx")
GSWForm.AddComboBoxItem(nComboBox, "UPS")
GSWForm.AddComboBoxItem(nComboBox, "USPS")
GSWForm.AddComboBoxItem(nComboBox, "DHL")

GSWForm.SetComboBoxSelection(nComboBox, "DHL")
GSWForm.CreateLabel(37, 2, 50, 3, "Carrier:")
```

AddComboBoxItem

This method adds an item to the ComboBox list.

GSWForm.**AddComboBoxItem**(i*Id As Integer, strText As String*)

PARAMETERS

iId

Required. **Integer** that specifies the ComboBox to add the item. The Identifier is returned from the CreateComboBox method.

strText

Required. **String** value that specifies the text of the item to add to the ComboBox.

RETURN VALUES

None

REMARKS

EXAMPLE

The following code segment shows how to add items to a ComboBox using the **AddComboBoxItem** method.

```
Dim nComboBox As Integer
nComboBox = GSWForm.CreateComboBox(38, 3, 52, 4)

GSWForm.AddComboBoxItem(nComboBox, "FedEx")
GSWForm.AddComboBoxItem(nComboBox, "UPS")
GSWForm.AddComboBoxItem(nComboBox, "USPS")
GSWForm.AddComboBoxItem(nComboBox, "DHL")

GSWForm.SetComboBoxSelection(nComboBox, "DHL")
GSWForm.CreateLabel(37, 2, 50, 3, "Carrier:")
```

SetComboBoxSelection

This method sets/initializes the value of the ComboBox

GSWForm.**SetComboBoxSelection**(i*Id As Integer, strText As String*)

PARAMETERS

iId

Required. **Integer** that specifies the ComboBox to perform the SetComboBoxSelection method. The Identifier is returned from the CreateComboBox method.

strText

Required. **String** value that specifies the text to use as the initial value for the ComboBox.

RETURN VALUES

None

REMARKS

EXAMPLE

The following code segment shows how to set the initial selection for a ComboBox using the **SetComboBoxSelection** method.

```
Dim nComboBox As Integer
nComboBox = GSWForm.CreateComboBox(38, 3, 52, 4)

GSWForm.AddComboBoxItem(nComboBox, "FedEx")
GSWForm.AddComboBoxItem(nComboBox, "UPS")
GSWForm.AddComboBoxItem(nComboBox, "USPS")
GSWForm.AddComboBoxItem(nComboBox, "DHL")

GSWForm.SetComboBoxSelection(nComboBox, "DHL")
GSWForm.CreateLabel(37, 2, 50, 3, "Carrier:")
```

GetComboBoxSelection

This method gets/reads the value of the ComboBox

GSWForm.**GetComboBoxSelection**(i*Id As Integer*)As String

PARAMETERS

iId

Required. **Integer** that specifies the ComboBox to perform the GetComboBoxSelection method. The Identifier is returned from the CreateComboBox method.

RETURN VALUES

The **String** value of the item selected in the ComboBox

REMARKS

EXAMPLE

The following code segment shows how to get the value of the ComboBox using the **GetComboBoxSelection** method.

```
Dim nComboBox As Integer
nComboBox = GSWForm.CreateComboBox(38, 3, 52, 4)

GSWForm.AddComboBoxItem(nComboBox, "FedEx")
GSWForm.AddComboBoxItem(nComboBox, "UPS")
GSWForm.AddComboBoxItem(nComboBox, "USPS")
GSWForm.AddComboBoxItem(nComboBox, "DHL")

GSWForm.SetComboBoxSelection(nComboBox, "DHL")
GSWForm.CreateLabel(37, 2, 50, 3, "Carrier:")

Dim nFormResult As Integer
nFormResult = GSWForm.Run()
If nFormResult = GSWForms.ButtonIDs.eIDCANCEL Then Return

'Get combo box selection text
Dim strComboBoxSelection As String
strComboBoxSelection = GSWForm.GetComboBoxSelection(nComboBox)
```

.

DropList Operations

The DropList is a type of graphical interface that allows the user to select exactly one of a predefined list. The DropList is useful when you do not want to permanently dedicate screen space to display the list contents. The drop list only displays the list when the down arrow is clicked. This is the difference in the DropList control and the List Box control (see page 52).

When the down arrow is clicked the entire list is displayed. If necessary the list may scroll if the number of items is greater than what will fit into the defined area.

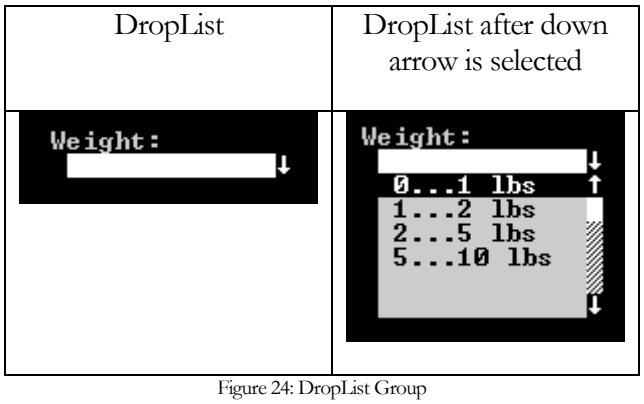

Figure 24: DropList Group

The DropList control displays a vertical list of items where the user can select exactly one item from the list. The developer may initialize an item as selected. The DropList returns a string that contains the text of the item selected.

DropList Control Functions		Description
CreateDropList		Creates and defines the location and size of the DropList
AddDropListItem		Adds items to the DropList
SetDropListSelection		Sets/Initializes an item in the DropList
GetDropListSelection		Reads the value of the DropList

Table 10 - DropList control functions

CreateDropList

This method creates and places a DropList on the GSWForm

```
ConsoleForm.CreateDropList(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow As Short)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of DropList.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of DropList.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of DropList.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of DropList.

RETURN VALUES

Returns an **Integer** that is used to identify this DropList when other methods are performed.

REMARKS

EXAMPLE

The following code segment shows how to create a DropList using the **CreateDropList** method.

```
Dim nDropList As Integer
nDropList = GSWForm.CreateDropList(38, 6, 52, 7)
GSWForm.AddDropListItem(nDropList, "0...1 lbs")
GSWForm.AddDropListItem(nDropList, "1...2 lbs")
GSWForm.AddDropListItem(nDropList, "2...5 lbs")
GSWForm.AddDropListItem(nDropList, "5...10 lbs")
GSWForm.CreateLabel(37, 5, 50, 6, "Weight:")
```

AddDropListItem

This method adds an item to the DropList

ConsoleForm.**AddDropListItem**(*iId As Integer, strText As String*)

PARAMETERS

iId

Required. **Integer** that specifies the DropList to add the item. The Identifier is returned from the CreateDropList method.

txtString

Required. **String** value that specifies the text of the item to add.

RETURN VALUES

None

REMARKS

EXAMPLE

The following code segment shows how to add items to a DropList using the **AddDropListItem** method.

```
Dim nDropList As Integer
nDropList = GSWForm.CreateDropList(38, 6, 52, 7)
GSWForm.AddDropListItem(nDropList, "0...1 lbs")
GSWForm.AddDropListItem(nDropList, "1...2 lbs")
GSWForm.AddDropListItem(nDropList, "2...5 lbs")
GSWForm.AddDropListItem(nDropList, "5...10 lbs")
GSWForm.CreateLabel(37, 5, 50, 6, "Weight:")
```

SetDropListSelection

This method sets/initializes the value of the DropList

GSWForm.**SetDropListSelection**(i*Id As Integer, strText As String*)

PARAMETERS

iId
Required. **Integer** that specifies the DropList to perform the SetDropListSelection method. The Identifier is returned from the CreateDropList method.

strText
Required. **String** value that specifies the text to use as the initial value for the DropList.

RETURN VALUES

None

REMARKS

EXAMPLE

The following code segment shows how to set initial DropList selection using the **SetDropListSelection** method.

```
Dim nDropList As Integer
nDropList = GSWForm.CreateDropList(38, 6, 52, 7)
GSWForm.AddDropListItem(nDropList, "0...1 lbs")
GSWForm.AddDropListItem(nDropList, "1...2 lbs")
GSWForm.AddDropListItem(nDropList, "2...5 lbs")
GSWForm.AddDropListItem(nDropList, "5...10 lbs")
GSWForm.SetDropListSelection(nDropList, "0...1 lbs")
GSWForm.CreateLabel(37, 5, 50, 6, "Weight:")
```

GetDropListSelection

This method gets/reads the value of the DropList

```
GSWForm.GetDropListSelection(iId As Integer)As String
```

PARAMETERS

iId

Required. **Integer** that specifies the DropList to perform the GetDropListSelection method. The Identifier is returned from the CreateDropList method.

RETURN VALUES

The **String** value of the selected item

REMARKS

EXAMPLES

The following code segment shows how to get the value of the DropList using the **GetDropListSelection** method.

```
Dim nDropList As Integer
nDropList = GSWForm.CreateDropList(38, 6, 52, 7)
GSWForm.AddDropListItem(nDropList, "0...1 lbs")
GSWForm.AddDropListItem(nDropList, "1...2 lbs")
GSWForm.AddDropListItem(nDropList, "2...5 lbs")
GSWForm.AddDropListItem(nDropList, "5...10 lbs")
GSWForm.SetDropListSelection(nDropList, 2)
GSWForm.CreateLabel(37, 5, 50, 6, "Weight:")

Dim nFormResult As Integer
nFormResult = GSWForm.Run()

If nFormResult = GSWForms.ButtonIDs.eIDCANCEL Then Return
Dim strDropListSelection As String

strDropListSelection = GSWForm.GetDropListSelection(nDropList)
```

EditBox Operations

The EditBox control provides an edit box that is available for a user input value and/or the display of a value. Most often it is a single line of text. Scroll arrows are displayed at the ends of box if the text length exceeds the length of the size of the displayed edit box. Validation templates are also available for quick data format verification. The EditBox can be configured so that a "TAB" or "ENTER" is automatically entered when the max EditBox Text Length is input

EditBox Control Functions		Description
CreateEditBox		Creates and defines the location of the EditBox Configure maximum text length and Automatic Tab
GetEditBoxText		Reads value from EditBox
SetEditBoxText		Sets the contents of the EditBox
SetEditBoxValidationFormat		Enforces correct data entry conforming to a specified validation template
SetEditBoxValidationRange		Enforces correct data entry within a specified rational number range
SetEditBoxValidationCreditCard		Enforces correct Credit Card data entry and validation conforming to well known standards

Table 11 - EditBox Control Functions

CreateEditBox

This method creates and defines the location of the EditBox. The CreateEditBox method also specifies the maximum text length allowed, the enabling of the Automatic Tab and returns and Identifier for further operations creates and places a EditBox on the GSWForm

```
ConsoleForm.CreateEditBox(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow as Short,IMaxTextLen as Integer,
IOptions As GSWForms.EditBoxOptions)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of EditBox.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of EditBox.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of EditBox.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of EditBox.

IMaxTextLen
Required. **Integer** that specifies the max text length that can be entered.
The TextLength input parameter determines the amount of text that can be entered into the edit box. If the input text is larger than the displayed size of the EditBox then scroll arrows appear at the ends of the EditBox.

IOptions
Required. **GSWForms.EditBoxOptions** that specifies if an Automatic Tab occurs when the IMaxTextLen is reached.

- GSWForms.EditBoxOptions.eAcceptDefaults
  - Normal Non Automatic Tab Operation
- GSWForms.EditBoxOptions.eAutoTab
  – Enable Automatic Tab.
  When the input to the EditBox reaches the specified TextLength, a TAB is automatically inserted moving to the next control.

RETURN VALUES

Returns an **Integer** that is used to identify this EditBox when other methods are performed.

EXAMPLE

The following code segment shows how to create an EditBox using the **CreateEditBox** method.

```
Dim nEditBox1 As Integer
nEditBox1 = GSWForm.CreateEditBox(3, 8, 34, 9, 50, _
                                  GSWForms.EditBoxOptions.eAcceptDefaults)
```

SetEditBoxText

This method sets/initializes the value of the EditBox

GSWForm.**SetEditBoxText**(i*Id As Integer, strText As String*)

PARAMETERS

iId

Required. **Integer** that specifies the EditBox to perform the SetEditBoxText method. The Identifier is returned from the CreateEditBox method.

strText

Required. **String** value that specifies the text to use as the initial value for the EditBox.

RETURN VALUES

None

REMARKS

EXAMPLE

The following code segment shows how to set an initial value in the EditBox using the **SetEditBoxText** method.

```
Dim nEditBox1 As Integer
GSWForm.SetEditBoxText(nEditBox1, "123 Webster Dr")
```

SetEditBoxValidationRange

This method enforces correct data entry within (inclusive) a specified rational number range.

GSWForm.**SetEditBoxValidationRange**(*iId As Integer, iMin as Integer, iMax As Integer)*

PARAMETERS

iId

Required. **Integer** that specifies the EditBox to perform the SetEditBox method. The Identifier is returned from the CreateEditBox method.

iMin

Required. **Integer** that specifies the ***minimum*** number that can be entered in the EditBox.

iMax

Required. **Integer** that specifies the **maximum** number that can be entered in the EditBox.

RETURN VALUES

None

REMARKS

If a value is entered that is either lower than the minimum or higher than the maximum value when the form is submitted RF FormMaker will automatically display a message box to the user indicating that the value does not fall into a valid range. Upon confirming the message box the user is returned back to the form.

EXAMPLE

The following code segment shows how to set the validation range for the EditBox using the **SetEditBoxValidationRange** method.

```
Dim nEditBox2 As Integer
GSWForm.SetEditBoxValidationRange(nEditBox2, -20, 590)
```

SetEditBoxValidationFormat

This method enforces correct data entry conforming to a specified validation template.

```
GSWForm.SetEditBoxValidationFormat(iId As Integer, iFormat As
GSWForms.ValidationFormats)
```

PARAMETERS

iId

Required. **Integer** that specifies the EditBox to perform the SetValidationFormat method. The Identifier is returned from the CreateEditBox method.

iFormat

Required. **GSWForms.ValidationFormats** that specifies one of the templates to use upon entry and validation of the user input to the EditBox.

VALIDATION TEMPLATES:

DATES	*Valid Input Examples*
o *eDate_DD_MM_YY*	*12/01/06*
o *eDate_DD_MM_YY_Opt_YY*	*12/01/06 or 12/01/2006*
o *eDate_DD_MM_YYYY*	*12/01/2006*
o *eDate_MM_DD_YY*	*01/12/2006*
o *eDate_MM_DD_YY_Opt_YY*	*01/12/06 or 01/12/2006*
o *eDate_MM_DD_YYYY*	*01/12/2006*
o *eDate_MM_YY*	*01/06*
o *eDate_MM_YY_Opt_YY*	*01/06 or 01/2006*
o *eDate_MM_YYYY*	*01/2006*

DATE OF BIRTH	Valid Input Examples
o *eDOB_DD_MM_YYYY*	*04/11/2005*
o *eDOB_MM_DD_YYYY*	*11/04/2005*
o *eDOB_MM_YYYY*	*11/2005*

TIME	Valid Input Examples
o *eTime_HH24_MM*	*16:40*
o *eTime_HH24_MM_Opt_SS*	*16:40 or 16:40:59*
o *eTime_HH24_MM_SS*	*16:40:59*
o *eTime_HH_MM_AM_PM*	*04:40 PM*
o *eTime_HH_MM_Opt_SS_AM_PM*	*04:40 AM or 04:40:59 PM*
o eTime_HH_MM_SS_AM_PM	04:40:59 PM

US Telephone	Valid Input Examples
o *eUSPhone_No_AC*	*265-1018*
o *eUSPhone_Opt_AC*	*265-1018 or (706) 265-1918*
o *eUSPhone_With_AC*	*(706) 265-1018*

SPECIAL VALIDATORS	Valid Input Examples
o *eSSN (Social Security Number)*	*345-69-1929*

RETURN VALUES

None

REMARKS

The template is used during input to the EditBox by automatically filling parts of the entry. Additionally, incorrect data types are not allowed for input. This helps to significantly reduce input errors by the user.

For example we know that when entering a social security number, we use the common format of three numbers followed by a dash, followed by two numbers, followed by a dash, followed by four numbers (sss-ss-ssss).

Using the eSSN validation template, after the first three numbers are entered RFFormMaker automatically inputs the dash '-' character. After the next two numbers are entered again RF FormMaker automatically enters the next dash '-' and then the next four numbers can be input.

If the user submitted the form after only entering two of the last four numbers then RF FormMaker would present the user with a message box indicating that the input was not valid allowing the user to correct the error.

EXAMPLE

The following code segment shows how to set the SSN validation template for SSN for the EditBox using the **SetEditBoxValidatationFormat** method.

```
Dim nEditBox2 As Integer
nEditBox2 = GSWForm.CreateEditBox(3, 11, 34, 12, 20, _
                                  GSWForms.EditBoxOptions.eAutoTab)
GSWForm.SetEditBoxValidationFormat(nEditBox2, _
                                  GSWForms.ValidationFormats.eSSN)
```

SetEditBoxValidationCreditCard

This method enforces correct data entry conforming to a specified credit card validation template.

GSWForm.**SetEditBoxValidationCreditCard**(*iId As Integer, iAllowedCards As GSWForms.CreditCards)*

PARAMETERS

iId

Required. **Integer** that specifies the EditBox to perform the SetValidationCreditCard method. The Identifier is returned from the CreateEditBox method.

iFormat

Required. **GSWForms.CreditCards** that specifies one of the Credit Card validation templates to use for entry and validation of the user input to the EditBox. Credit card templates can be 'Or' ed using the Visual Basic **bitwise** 'Or' operator. For example you may take Visa OR AmEx.

VALIDATION TEMPLATES:

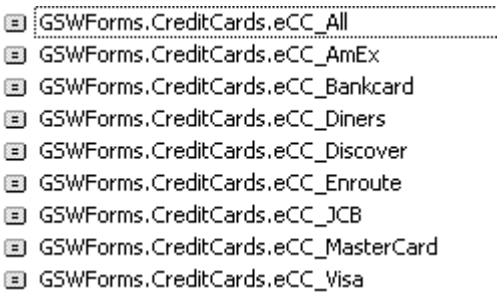

Figure 25: Credit Card Validation Templates

RETURN VALUES

REMARKS

The Credit Card Validation templates enforce common standards for credit card number entry. This includes card number lengths, required values for certain positions, etc.

EXAMPLE

The following code segment shows how to set the CreditCard validation template for Visa for the EditBox using the **SetEditBoxValidationCreditCard** method.

```
Dim nEditBox2 As Integer
nEditBox2 = GSWForm.CreateEditBox(3, 11, 34, 12, 20, _
            GSWForms.EditBoxOptions.eAutoTab)
GSWForm.SetEditBoxValidationCreditCard(nEditBox2, _
            GSWForms.CreditCards.eCC_Visa)
```

GetEditBoxText

This method reads/gets the value of the edit box.

ConsoleForm.**GetEditBoxText**(i*Id As Integer) As String*

PARAMETERS

iId

Required. **Integer** that specifies the EditBox to perform the SetEditBox method. The Identifier is returned from the CreateEditBox method.

RETURN VALUES

strText

A **String** that contains the EditBox Value.

EXAMPLE

```
Dim strValue as String
strValue = GSWForm.GetEditBoxText(nEditBox2)
```

Label Operations

The Label control creates and positions a text description on the form. With RF FormMaker the color of the text and the text background can be specified providing pleasant visual distinction.

Label Control Functions		Description
CreateLabel		Creates and defines the location and size of the DropList
SetLabelColor		Adds items to the DropList

Table 12 – Label Control Operations

CreateLabel

This method creates and defines the location of the Label.

```
ConsoleForm.CreateLabel(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow as Short, strText as String)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of Label.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of Label.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of Label.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of Label.

strText
Required. **String** that specifies the text of the Label.

RETURN VALUES

Returns an **Integer** that is used to identify this Label when other methods are performed.

EXAMPLE

```
Dim nLabel As Integer = GSWForm.CreateLabel(3, 13, 34, 15, _
      "Press OK to accept selections or Cancel to abort changes")
GSWForm.SetLabelColor(nLabel, GSWForms.BackgroundColors.eBkgWhite, _
                              GSWForms.ForegroundColors.eFgBrightRed)
```

SetLabelColor

This method sets the foreground and background colors for the label text.

```
ConsoleForm.SetLabelColor(iId As Integer, Background As GSWForms.BackgroundColors,
Foreground As GSWForms.ForegroundColors)
```

PARAMETERS

iID

Required. **Integer** that specifies the Label to perform the SetLabelColor method. The Identifier is returned from the CreateLabel method

Background

Required. **GSWForms.BackgroundColors** is an enumerated list were valid values can be selected from the following list.

Figure 26: Label Background Colors

Foreground

Required. **GSWForms.ForegroundColors** is an enumerated list were valid values can be selected from the following list.

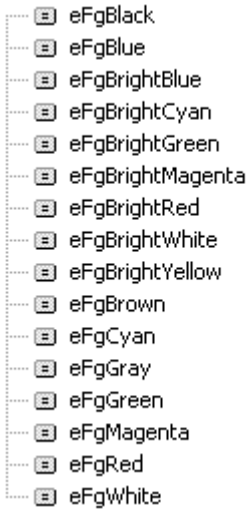

Figure 27: Label Foreground Colors

RETURN VALUES

EXAMPLE

```
Dim nLabel As Integer = GSWForm.CreateLabel(3, 13, 34, 15, _
      "Press OK to accept selections or Cancel to abort changes")
GSWForm.SetLabelColor(nLabel, GSWForms.BackgroundColors.eBkgWhite, _
                              GSWForms.ForegroundColors.eFgBrightRed)
```

ListBox Operations

The ListBox control is a type of graphical interface that allows the user to select one of a predefined list. The list may scroll if the number of items is greater than what will fit into the defined area.

Figure 28: ListBox Group

The ListBox control displays a vertical list of items where the user can select exactly one item from the list. The developer may initialize an item as selected. The ListBox control will return the numeric (Index-1) of item selected or the index number of the item selected. In the example above Red would be a 0, Blue a 1, Green a 2 and so on.

ListBox Control Functions		Description
CreateListBox		Creates and defines the location and size of the ListBox
AddListBoxItem		Adds items to the ListBox
SetListBoxSelection		Sets/Initializes an item in the ListBox
GetListBoxSelection		Reads the value of the ListBox.

Table 13 - ListBox control functions

CreateListBox

This method creates and places a ListBox on the GSWForm.

```
ConsoleForm.CreateListBox(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow As Short)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of ListBox.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of ListBox.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of ListBox.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of ListBox.

RETURN VALUES

Returns an **Integer** that is used to identify this ListBox that can is used when other methods are performed.

REMARKS

EXAMPLE

```
Dim nListBox As Integer
nListBox = GSWForm.CreateListBox(15, 3, 34, 7)
```

AddListBoxItem

This method adds an item to the ListBox

ConsoleForm.**AddListBoxItem**(*iId As Integer, strText As String*)

PARAMETERS

Identifier

Required. **Integer** that specifies the ListBox to add the item. The Identifier is returned from the CreateListBox method.

txtString

Required. **String** value that specifies the text of the item to add to the ListBox.

RETURN VALUES

None

REMARKS

EXAMPLE

```
Dim nListBox As Integer
nListBox = GSWForm.CreateListBox(38, 9, 52, 17)
GSWForm.AddListBoxItem(nListBox, "Red")
GSWForm.AddListBoxItem(nListBox, "Blue")
GSWForm.AddListBoxItem(nListBox, "Green")
GSWForm.AddListBoxItem(nListBox, "Magenta")
GSWForm.AddListBoxItem(nListBox, "Cyan")
GSWForm.AddListBoxItem(nListBox, "Yellow")
GSWForm.AddListBoxItem(nListBox, "Pink")
GSWForm.AddListBoxItem(nListBox, "Black")
GSWForm.AddListBoxItem(nListBox, "White")
GSWForm.AddListBoxItem(nListBox, "Orange")
GSWForm.SetListBoxSelection(nListBox, 2)
GSWForm.CreateLabel(37, 8, 50, 9, "Avail. colors:")
```

SetListBoxtSelection

This method sets/initializes the value of the ListBox.

GSWForm.**SetListBoxSelection**(i*Id As Integer, iSelection As Integer*)

PARAMETERS

iId

Required. **Integer** that specifies the ListBox to perform the SetListBoxSelection method. The Identifier is returned from the CreateListBox method.

iSelection

Required. **Integer** that specifies the index of the item to select in the ListBox as the initial value. Zero based. First item is 0, Second is 1, and so on.

RETURN VALUES

None

REMARKS

EXAMPLES

The following code segment shows how to initialize an item to be the initial selection using the **SetListBoxSelection** method.

```
Dim nListBox As Integer
nListBox = GSWForm.CreateListBox(38, 9, 52, 17)
GSWForm.AddListBoxItem(nListBox, "Red")
GSWForm.AddListBoxItem(nListBox, "Blue")
GSWForm.AddListBoxItem(nListBox, "Green")
GSWForm.AddListBoxItem(nListBox, "Magenta")
GSWForm.AddListBoxItem(nListBox, "Cyan")
GSWForm.AddListBoxItem(nListBox, "Yellow")
GSWForm.AddListBoxItem(nListBox, "Pink")
GSWForm.AddListBoxItem(nListBox, "Black")
GSWForm.AddListBoxItem(nListBox, "White")
GSWForm.AddListBoxItem(nListBox, "Orange")
GSWForm.SetListBoxSelection(nListBox, 2)
GSWForm.CreateLabel(37, 8, 50, 9, "Avail. colors:")
```

GetListBoxtSelection

This method gets/reads the value of the ListBox.

GSWForm.**GetListBoxSelection**(i*Id As Integer*) As Integer

PARAMETERS

iId

Required. **Integer** that specifies the ListBox to perform the SetListBoxSelection method. The Identifier is returned from the CreateListBox method.

RETURN VALUES

Integer that specifies the index of the item selected in the ListBox. Zero based. First item is 0, Second is 1, and so on.

REMARKS

EXAMPLE

The following code segment shows how to read the selected item using the GetListBoxSelection method

```
Dim nListBoxSelection As Integer
nListBoxSelection = GSWForm.GetListBoxSelection(nListBox)
```

MessageBoxA Operations

This method provides a quick and easy way to display a message box on the ConsoleScreen. A message is displayed to the user and a OK button is automatically provided to dismiss the messagebox returning the user back to the original form.

```
GSWForm.MessageBoxA(strText As String)
```

PARAMETERS

strText

Required. **String** that specifies the text message to display in the MessageBox.

RETURN VALUES

REMARKS

EXAMPLE

```
GSWForm.MessageBoxA("PO number not in the database!")
```

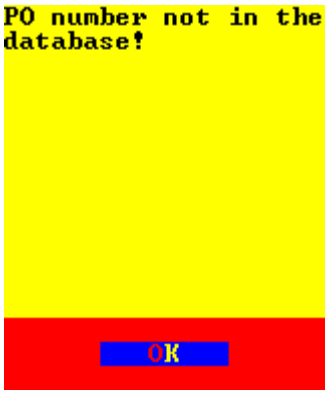

Figure 29: MessageBoxA

RadioButton Operations

The Radio Button control is a type of graphical interface that allows the user to select one of the predefined set of options.

Figure 30: Radio Button Group

The Radio Button group control displays a vertical list of options where the user can select exactly one item from the list. The developer may initialize an item as selected or leave no items selected. The RadioButtonGroupSelection will return a zero if no item was selected or the index number of the item selected.

RadioButtonControl Functions		Description
CreateRadioButtonGroup		Creates and defines the location and size of the Radio Button Group
AddRadioButtonGroupItem		Adds items to the list of Radio Button group options
SetRadioGroupSelection		Sets/Initializes an item in the RadioButtonGroup
GetRadioGroupSelection		Reads the value of the RadioButtonGroup.

Table 14 - RadioButtonGroup control functions

CreateRadioButtonGroup

This method creates and places a Radio Button Group on the GSWForm

```
ConsoleForm.CreateRadioButtonGroup(usTopLeftCol As Short,usTopLeftRow As Short,
usBottomRightCol As Short, usBottomRightRow As Short)As Integer
```

PARAMETERS

usTopLeftCol
Required. **Short** that specifies the Cell Coordinates of top left column of Radio Button Group.

usTopLeftRow
Required. **Short** that specifies the Cell Coordinates of top left row of Radio Button Group.

usBottomRightCol
Required. **Short** that specifies the Cell Coordinates of bottom right column of Radio Button Group.

usBottomRightRow
Required. **Short** that specifies the Cell Coordinates of bottom right row of Radio Button Group.

RETURN VALUES

REMARKS

EXAMPLE

```
Dim nRadioButtonGroup As Integer
nRadioButtonGroup = GSWForm.CreateRadioButtonGroup(15, 3, 34, 7)
```

Note: When specifying the location and size of the RadioButtonGroup be sure to make it large enough to contain all the option items. If it is not large enough then those options will not be displayed.

AddRadioButtonGroupItem

This method adds an item to the RadioButtonGroup.

ConsoleForm.**AddRadioButtonGroupItem**(*iId As Integer, strText As String*)

PARAMETERS

Identifier

Required. **Integer** that specifies the Radio Button Group to add the item. The Identifier is returned from the CreateRadioButtonGroup method.

txtString

Required. **String** value that specifies the text of the item to add to the Radio Button Group. The text is displayed to the right of the Radio Button.

RETURN VALUES

None

REMARKS

EXAMPLE

```
Dim nRadioButtonGroup As Integer
nRadioButtonGroup = GSWForm.CreateRadioButtonGroup(15, 3, 34, 6)
GSWForm.AddRadioButtonGroupItem(nRadioButtonGroup, "~N~orth America")
GSWForm.AddRadioButtonGroupItem(nRadioButtonGroup, "~E~urope")
GSWForm.AddRadioButtonGroupItem(nRadioButtonGroup, "~A~sia")
```

SetRadioButtonGroupSelection

This method sets/initializes the value of the Radio Button Group.

GSWForm.**SetRadioButtonGroupSelection**(i*Id As Integer, iSelection As Integer*)

PARAMETERS

iId

Required. **Integer** that specifies the Radio Button Group to perform the SetRadioButtonGroupSelection method. The Identifier is returned from the CreateRadioButtonGroup method.

iSelection

Required. **Integer** that specifies the index of the item to select in the Radio Button Group as the initial value. Zero is no item is selected. First item is 1, Second is 2, and so on.

RETURN VALUES

None

REMARKS

EXAMPLE

```
Dim nRadioButtonGroup As Integer
nRadioButtonGroup = GSWForm.CreateRadioButtonGroup(15, 3, 34, 6)
GSWForm.AddRadioButtonGroupItem(nRadioButtonGroup, "~N~orth America")
GSWForm.AddRadioButtonGroupItem(nRadioButtonGroup, "~E~urope")
GSWForm.AddRadioButtonGroupItem(nRadioButtonGroup, "~A~sia")
'0 - nothing selected, 1 - first item etc.
GSWForm.SetRadioButtonGroupSelection(nRadioButtonGroup, 2)
```

Note: The 3rd item in the Radio Button Group is set. See Figure 30: Radio Button Group to see how it is displayed.

GetRadioButtonGroupSelection

This method gets/reads the value of the RadioButtonGroup.

GSWForm.**GetRadioButtonGroupSelection**(i*Id As Integer*) As Integer

PARAMETERS

iId

Required. **Integer** that specifies the Radio Button Group to perform the GetRadioButtonGroupSelection method. The Identifier is returned from the CreateRadioButtonGroup method.

RETURN VALUES

Integer that specifies the index of the item selected in the Radio Button Group. Zero if no items are selected. First item is 1, Second item is 2 and so on.

REMARKS

EXAMPLE

The following code segment shows how to read the selected item using the GetRadioButtonSelection method

```
Dim nRadioIndex As Integer
nRadioIndex = GSWForm.GetRadioButtonGroupSelection(nRadioButtonGroup)
```

Run Method

The Run method displays the Form to the user. It executes the Form, creating and placing all the controls as specified using the Create, ADD and Set Methods for each control. Additionally the Run method places the Form on the ConsoleScreen so that it is displayed to the user. RF FormMaker takes care of all the user interaction associated with text entry, selection, checkboxes, radio buttons etc., and returns program control when either a Button is clicked or the ESC is entered.

When program control is returned the developer can retrieve the values for each control on the form and continue program execution as needed.

```
GSWForm.Run() As Integer
```

PARAMETERS

RETURN VALUES

Integer that specifies the value of the button clicked to exit the form. This value is the iCommand parameter specified for the button during the create button method.

Possible values are:

- eIDCANCEL
- eIDNO
- eIDOK
- eIDYES
- Value in the range from eIDUSERMAX to eIDUSERMIN. See section on Create Button on page 23.

REMARKS

If the ESCAPE key is used to exit the form then the return value is eIDCANCEL

EXAMPLE

The following code segment shows how display the form using the Run method.

```
Dim nFormResult As Integer
'This will display the form
nFormResult = GSWForm.Run()
If nFormResult = GSWForms.ButtonIDs.eIDCANCEL Then
    Return
End If
```

UTS Object

Many parameters of the GSW UTS session are available through the UTS Control. The UTS object is only available with the Runtime license. Additionally the Beep method can be used to send the beep sound to the client device.

Properties		Description
GSWClient		Set to 1 or 0 depending if a GSW or 3rd Party client is connected. If GSW then the value is 1 otherwise the value is 0
iServerPort		Port associated with the session
iSessionId		Process id of the Agent Process handling the user session
iUseColor		The Color or Monochrome presentation for 3rd party clients
strAnswerback		Answerback text passed from the GSW Client
strClientIP		IP address of the client computer/device
strClientMAC		MAC (Media Access Control) address of the client computer/device
strTTY		The Georgia SoftWorks Server creates a tty name on a per session basis

Table 15 - UTS Properties

Methods		Description
Beep		Sends the Beep sound to the device

Table 16 - UTS Methods

Beep

This method causes the Beep sound to occur on the device a specified number of times.

```
GSWForms.UTSAccess.Beep(iCount As Integer)
```

PARAMETERS

iCount

Required. **Integer** that specifies the number of times the Beep sound will occur.

RETURN VALUES

REMARKS

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
Try
    UTSAccess = New GSWForms.UTS
Catch ex As Exception
    Console.WriteLine("Unable to create GSWForms.UTS object: <" &
    ex.Message & ">")
    Return
 End Try

 UTSAccess.Beep(2)
```

Client MAC Address

This property gets the Client MAC (Media Access Control) address.

GSWForms.UTSAccess.**strClientMAC**

.

RETURN VALUES

String that contains the Client MAC address.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
  Try
      UTSAccess = New GSWForms.UTS
  Catch ex As Exception
      Console.WriteLine("Unable to create GSWForms.UTS object: <" &
      ex.Message & ">")
      Return
   End Try

   GSWForm.CreateLabel(1, 0, 15, 1, "strClientMAC: ")
   GSWForm.CreateLabel(18, 0, 38, 1, UTSAccess.strClientMAC)
```

Client IP Address

This property returns the IP Address of the Client.

GSWForms.UTSAccess.**strClientIP**

RETURN VALUES

String that contains the IP Address of the client.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
  Try
      UTSAccess = New GSWForms.UTS
  Catch ex As Exception
      Console.WriteLine("Unable to create GSWForms.UTS object: <" &
      ex.Message & ">")
      Return
  End Try
  GSWForm.CreateLabel(1, 6, 15, 7, "strAnswerback:")
  GSWForm.CreateLabel(18, 7, 38, 8, UTSAccess.strClientIP)
```

Server Port Address

This property returns the port associated with the session.

GSWForms.UTSAccess.**iServerPort**

RETURN VALUES

Integer that contains the port associated with the session.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
  Try
      UTSAccess = New GSWForms.UTS
  Catch ex As Exception
      Console.WriteLine("Unable to create GSWForms.UTS object: <" &
      ex.Message & ">")
      Return
   End Try

   GSWForm.CreateLabel(1, 1, 15, 2, "lServerPort:  ")
   GSWForm.CreateLabel(18, 0, 38, 1, UTSAccess.lServerPort)
```

Session ID

This property returns the process ID associated with the session.

GSWForms.UTSAccess.**iSessionId**

RETURN VALUES

Integer that contains the process id associated with the session.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
  Try
      UTSAccess = New GSWForms.UTS
  Catch ex As Exception
      Console.WriteLine("Unable to create GSWForms.UTS object: <" &
      ex.Message & ">")
      Return
   End Try

   GSWForm.CreateLabel(1, 1, 15, 2, "lSessionID:  ")
   GSWForm.CreateLabel(18, 0, 38, 1, UTSAccess.lSessionID)
```

TTY Name

This property returns the TTY Name on a per session basis.

GSWForms.UTSAccess.**strTTY**

RETURN VALUES

STRING that contains the TTY Name for the session.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
Try
    UTSAccess = New GSWForms.UTS
Catch ex As Exception
    Console.WriteLine("Unable to create GSWForms.UTS object: <" & _
    ex.Message & ">")
    Return
 End Try

 GSWForm.CreateLabel(1, 7, 15, 8, "strClientIP:  ")
 GSWForm.CreateLabel(18, 0, 38, 1, UTSAccess.strTTY)
```

Use Color

This property get the configured value that indicates whether to use color or monochrome.

GSWForms.UTSAccess.**iUseColor**

RETURN VALUES

Integer NonZero if color is used. Zero is monochrome.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
Try
    UTSAccess = New GSWForms.UTS
Catch ex As Exception
    Console.WriteLine("Unable to create GSWForms.UTS object: <" &
    ex.Message & ">")
    Return
End Try

GSWForm.CreateLabel(1, 5, 15, 6, "lUseColor:    ")
GSWForm.CreateLabel(18, 0, 38, 1, UTSAccess.iUseColor)
```

GSWClient

This property returns the value that indicates if the client on the device is a GSW Client or a 3rd Party client.

GSWForms.UTSAccess.**iGSWClient**

RETURN VALUES

Integer that indicates if the client is a GSW Client or a 3rd party client.

GSW Client = 1, 3rd Party Client = 0

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
  Try
      UTSAccess = New GSWForms.UTS
  Catch ex As Exception
      Console.WriteLine("Unable to create GSWForms.UTS object: <" &
      ex.Message & ">")
      Return
  End Try
  GSWForm.CreateLabel(1, 4, 15, 5, "GSWClient:    ")
  GSWForm.CreateLabel(18, 4, 38, 5, UTSAccess.GSWClient)
```

Answerback String

This property returns the Answerback string.

GSWForms.UTSAccess.**strAnswerback**

RETURN VALUES

String that contains the Answerback text.

EXAMPLE

```
Dim UTSAccess As GSWForms.UTS
  Try
      UTSAccess = New GSWForms.UTS
  Catch ex As Exception
      Console.WriteLine("Unable to create GSWForms.UTS object: <" &
      ex.Message & ">")
      Return
  End Try
  GSWForm.CreateLabel(1, 6, 15, 7, "strAnswerback:")
  GSWForm.CreateLabel(18, 6, 38, 7, UTSAccess.strAnswerback)
```

Error Handling

It is important to use good error handling practices. GSW recommends that you use the Try-Catch error handling technique in order that you can identify any problem that may occur. If an error occurs and you do not have an error handler in place it will try to be displayed on the server. This could be difficult to identify because the user will see their Telnet or SSH2 session hang.

Each method and each property retrieval may throw an exception due to programming errors, ActiveX or registration problems.

Exceptions thrown by GSW will always have a string parameter describing the problem which was encountered.

```
Dim UTSAccess As GSWForms.UTS
Try
    UTSAccess = New GSWForms.UTS
Catch ex As Exception
    Console.WriteLine("Unable to create GSWForms.UTS object: <" & _
    ex.Message & ">")
    Return
End Try
```

Examples

Several running examples are included on the CD and in the download.

RF FormMaker Controls

FormMaker Demo shows how to create a Form that uses all the controls.

Navigation Example

Shows how to create a navigation form which uses buttons to implement naviation.

DataBase Example

Shows how to interface with a Microsoft Access Database(JET).

UTS Example

Shows how to send beeps to client devices and read UTS properties.

Registration

The GSW RF FormMaker is licensed for a single server. The license must be *activated* for the software to operate. To activate the license a valid *Serial Number* is required.

Registration for the RF FormMaker is the same registration as for the UTS. Perform the **UTS registration** again and when the Product ID is sent to GSW request that the RF FormMaker be activated as well.

When GSW receives the request, the customer service attendant will return a new Serial Number that will activate RF FormMaker upon validation of a proper RF FormMaker purchase.

Please see UTS User Manual for registration procedures.

GSW RF FormMaker Subscription

Currently the GSW RF FormMaker inherits the subscription plan purchased with the UTS.

HOW TO UPDATE THE SOFTWARE

1. Simply download and install the new version of RF FormMaker.

HOW TO RENEW THE GSW RF FormMaker Subscription

Please use the following procedure when renewing the RF FormMaker Subscription.

Step	Who		Action
1.	GSW		Send notice to customer indicating that the subscription is about to expire. The notice is sent approximately 4 to 8 weeks prior to the expiration of the plan.
2.		Customer	Places order for new subscription
3.	GSW		Confirms Order
4.	GSW		Ships current software, documentation and new Floating License
5.		Customer	Install new Floating License (and software if desired)
6.		Customer	Ships OLD Floating License back to GSW

Table 17: Steps to Renew the GSW Subscription Plan

System Signature - IMPORTANT PLEASE READ

NOTE: This section only applies to Software Registration

The registration software obtains a system signature that is unique to your system. This signature is an added security measure to inhibit unauthorized personnel to obtain working copies of the GSW RF FormMaker .

The signature is comprised of hardware and software identifiers that exist on your system that make the target system unique. These identifies are hashed into a Product ID and a Serial Number can be generated from this Product id.

If major hardware components of your system are removed, replaced or modified your **Serial Number** may discontinue to work and you may need a new **Serial Number** to obtain access to the RF FormMaker. Please contact Georgia SoftWorks Technical Support if needed.

Technical Support

In order to keep Technical Support **Free** please help keep our cost down.

- Gather all relevant system and environment information.
- Write your question down. This not only helps us but also helps you in articulating the question.

Provide Log Files To GSW Technical Support

A typical sequence when GSW Technical Support needs the logs files are to delete the log files, reproduce the behavior in question and email the log files, which are recreated during the test, to GSW Support.

Email Support Tips:

To expedite support for suspected problems please perform the following test steps below to help us diagnose the issue.

1. Disconnect all users. Make sure that no other user connects at the time of the test.
2. Wait 5 minutes
3. Delete the Log files

   Delete all log files from the GSW UTS Server installation 'Log' subdirectory on the computer running the GSW Universal Terminal Server. (Usually `c:\GS_UTS\RF_FM.Log`)
4. To expedite resolution, reboot the Server if possible
5. Duplicate the problem.
6. The log files are automatically re-created. Send us the files in an email to support@georgiasoftworks.com
7. Please also include
   a. A description of the problem including User ID's, Domain and IP Addresses
   b. The logon script associated with the user experiencing the problem. (That is the `c_start.bat` or the `k_start.bat` file that resides in the scripts folder in the GSW UTS directory
   c. And of course your contact information.

If the question is not an emergency, please use e-mail at support@georgiasoftworks.com. We try to respond within 24 hours.

Or **Call 706.265.1018 EST, M-F 9:00 a.m. to 5:00 p.m. and have your Product ID ready**